

Minimizing Multimodular Functions and Allocating Capacity in Bike-Sharing Systems^{*}

Daniel Freund, Shane G. Henderson, and David B. Shmoys

Cornell University, Ithaca, NY, USA 14853
 {df365, sgh9, dbs10}@cornell.edu

Abstract. The growing popularity of bike-sharing systems around the world has motivated recent attention to models and algorithms for the effective operation of these systems. Most of this literature focuses on their daily operation for managing asymmetric demand. In this work, we consider the more strategic question of how to allocate dock-capacity in such systems. Our main result is a practically fast polynomial-time allocation algorithm to compute optimal solutions for this problem, that can also handle a number of practically motivated constraints, such as a limit on the number of docks moved from a given allocation. Our work further develops connections between bike-sharing models and the literature on discrete convex analysis and optimization.

1 Introduction

As shared vehicle systems, such as bike-sharing and car-sharing, become an integral part of urban landscapes, novel lines of research seek to model and optimize the operations of these systems. In many systems, such as New York City’s Citi Bike, users can rent and return bikes at any location throughout the city. This flexibility makes the system attractive for commuters and tourists alike. From an operational point of view, however, this flexibility leads to imbalances when demand is asymmetric as is commonly the case. The main result of this paper is to identify key questions in the design of operationally efficient bike-sharing systems, and to provide a polynomial algorithm for the associated discrete optimization problems.

Most bike-sharing systems are dock-based, meaning that they consist of stations, spread across the city, each of which has a number of docks in which bikes are locked. If a bike is present in a dock, users can rent it and return it at any other station with an open dock. However, system imbalance often causes some stations to have only empty docks and others to have only full docks. In the former case, users need to find alternate modes of transportation, whereas in the latter they might not be able to end their trip at the intended destination. In many bike-sharing systems, this has been found to be a leading cause of customer dissatisfaction (see e.g., [2]).

^{*} Work supported in part under NSF grants CCF-1526067, CMMI-1537394, CCF-1522054, and CMMI-1200315.

In order to meet demand in the face of asymmetric traffic, bike-sharing system operators seek to *rebalance* the system by moving bikes from locations with too few open docks to locations with too few bikes. To facilitate these operations, a burst of recent research has investigated models and algorithms to increase their efficiency and increase customer satisfaction. While similar in spirit to some of the literature on rebalancing, in this work we use a different control to increase customer satisfaction. Specifically, we answer the question *how should bike-sharing systems allocate dock capacity to stations within the system so as to minimize the number of dissatisfied customers?*

Related Work Raviv and Kolka [29] defined a *user dissatisfaction function* that measures the expected number of out-of-stock events at an individual bike-share station. To do so, they define a continuous-time Markov chain on the possible number of bikes (between 0 and the capacity of the station). Bikes are rented with rate $\lambda(t)$ and returned with rate $\mu(t)$. Each arrival triggers a change in the state, either decreasing (rental) or increasing (return) the number of available bikes by one. When the number of bikes is 0 and a rental occurs, the customer experiences an out-of-stock event. Using a discrete Markov Chain, they approximate the expected number of out-of-stock events over a finite time-horizon. For fixed rates, the work of Schuijbroek et al. [31] and O’Mahony [25] give different techniques to compute the expected number of out-of-stock events exactly. A recursion suggested by Parikh and Ukkusuri [27] shows that these methods extend to settings in which rates are constant over intervals.

The definition of the user dissatisfaction function triggered a line of work around static rebalancing problems, in which a capacitated truck (or a fleet of trucks) is routed over a limited time horizon. The truck may pick up and drop off bikes at each station, so as to minimize the expected number of out-of-stock events that occur after the completion of the route. Variations of this setting include Raviv et al. [30], Forma et al. [10], Kaspi et al. [18], Ho et al. [14], and Freund et al. [11]. As in our work, all of these papers make the assumption that demand is exogeneous and independent among stations, i.e., reducing the number of bikes available for rentals upstream has no effect on the number of returns downstream.

In contrast to the other papers mentioned, O’Mahony [25] addressed the question of allocating both docks and bikes; he uses the user dissatisfaction function to design a mixed integer program over the possible allocations of bikes and docks. In essence, our work extends upon this by providing a fast polynomial-time algorithm for that same problem and an extension thereof. The optimal allocation of bikes has also been studied by Jian and Henderson [16], Datner et al. [6], and by Jian et al. [15], with the latter also considering the allocation of docks. They each develop frameworks based on ideas from simulation optimization; while they also treat demand for bikes as being exogeneous, their framework captures the downstream effects of changes in supply upstream.

The work of Kaspi et al. [18] investigates the effects that broken bikes can have on the cost-function. Interestingly, some of their results can be viewed as analogous to ours. They prove that the cost-function at one station is M -

convex (see the book by Murota [22] and the references therein); surprisingly, the existing literature on the minimization of naturally M convex functions does not seem to capture the optimization problems we consider. We discuss the similarities and differences below.

Orthogonal approaches to the question of where to allocate docks have been taken by Kabra et al. [17] and Wang et al. [33]. The former considers demand as endogeneous and aims to identify the station density that maximizes sales, whereas we consider demand and station locations as exogeneously given and aim to allocate docks and bikes to maximize the amount of demand that is being met. The latter aims to use techniques from retail location theory to find locations for stations to be added to an existing system.

Further related work includes a line of work on rebalancing triggered by Chemla et al. [4]. Subsequent papers, e.g., by Nair et al. [24], Dell’Amico et al. [7], Erdogmus et al. [9], and [8], solve the routing problem with fixed numbers of bikes that need to be picked up/dropped off at each station – work by de Chardon et al. [3] surveys many of these works. Other approaches to rebalancing include for example the papers of Liu et al. [21], Ghosh et al. [12], and Rainer et al. [28]. While all of these fall into the wide range of recent work on the operation of bike-sharing systems, they differ from our work in the controls and methodologies they employ.

Finally, a great deal of work has been conducted in the context of predicting demand. In this work, we assume that the predicted demand is given, e.g., using the methods of O’Mahony and Shmoys [26] or Singhvi et al. [32]. Further methods to predict demand have been suggested by Li et al. [20], Chen et al. [5], and Zhang et al. [34] among others. Our results can be combined with any such works that predict demand at each station independently of all others.

Our Contribution We consider the problem of allocating dock capacity in bike-share systems in a setting in which we are given distributional knowledge about exogeneous demand at each station that is independent of our solution. This allows us, using techniques from [25], [31], and [27] to compute the expected number of out-of-stock events $c_i(d_i, b_i)$ at each station i for a given allocation of b_i bikes and d_i empty docks (i.e., $d_i + b_i$ docks in total).

Given this cost-function, we want to find the allocation of bikes and docks in the system that minimizes the total expected number of out-of-stock events within a system of n stations, i.e., $\sum_{i=1}^n c_i(d_i, b_i)$. However, due to the number of bikes and docks being limited, we need to accommodate a *budget constraint* on the number of bikes in the system and another on the number of docks in the system. Other constraints are often important, such as lower and upper bounds on the allocation for a particular station; furthermore, one important issue that has arisen in our collaboration with Citi Bike in NYC is that we seek to optimize the allocation while limiting the number of docks moved from the current system configuration. Our methods are amenable to these *operational constraints*. Finally, one additional type of constraint is that the allocation given to disjoint neighborhoods must provide equitable access to the system; this can be modeled through a laminar family of set constraints, and our techniques can

be extended to handle these by a standard dynamic programming approach, albeit with somewhat slower running times.

We design an algorithm that provably solves the minimization problem in $O(nT + (T + \log(n))(B + D))$ when given access to an oracle that computes $c_i(d, b)$ in $O(T)$ — [25] takes $O((d + b)^3)$. Our algorithm exploits the fact that the cost-function $c(\cdot, \cdot)$ is multimodular (cf. Definition 1) at each station.

The multimodularity provides an interesting connection to the literature on discrete convex analysis. Recent work [18] has shown independently that the number of out-of-stock events $F(b, U - d - b)$ at a bike-share station with fixed capacity U , b bikes and $U - d - b$ unusable bikes is M -natural convex in b and $U - d - b$. Unusable bikes effectively reduce the capacity at the station, since they are assumed to remain in the station over the entire time horizon. A station with capacity U , b bikes, and $U - b - d$ unusable bikes, must then have d empty docks; hence, $c(d, b) = F(b, U - d - b)$ for $d + b \leq U$, which parallels our result that $c(\cdot, \cdot)$ is multimodular. Though this would suggest that algorithms to minimize M -convex functions could solve our problem optimally, we show in Appendix 5 that M -convexity is not preserved, even in the version with only budget constraints.¹ However, since multimodularity is preserved we believe that techniques by Murota [23], combined with the submodular function minimization algorithms of Lee et al. [19], give a $O(n^3T + n^4(D + B))$ algorithm to solve the version with only budget constraints. By exploiting the separability of our objective function and the associated multimodularity of each station’s cost function, we obtain algorithms with significantly stronger running-time guarantees that quickly yield solutions for instances at the scales that typically arise in practice.

2 Model

We denote by $X = (X_1, \dots, X_s) \in \{\pm 1\}^s$ a sequence of s customers at a bike-share station. The sign of X_t identifies whether customer t arrives to rent or to return a bike, i.e., if $X_t = 1$ customer t wants to return a bike and if $X_t = -1$ customer t wants to rent a bike. The truncated sequence (X_1, \dots, X_t) is written as $X(t)$. We denote throughout by d and b the number of open docks and available bikes at a station before any customer has arrived. Notice that a station with d open docks and b available bikes has $d + b$ docks in total. Whenever a customer arrives to return a bike at a station and there is an open dock, the customer returns the bike, the number of available bikes increases by 1 and the number of open docks decreases by 1. Similarly, a customer arriving to rent a bike when one is available decreases the number of available bikes by 1 and increases the number of open docks by 1. If however a customer arrives to rent (return) a bike when no bike (open dock) is available, then she disappears with an out-of-stock event. We assume that only customers affect the inventory-level at a station, i.e., no rebalancing occurs. It is useful then to write

$$\delta_{X(t)}(d, b) := \max\{0, \min\{d + b, \delta_{X(t-1)} - X_t\}\}, \quad \delta_{X(0)}(d, b) = d$$

¹ Specifically, we (i) give an example in which a M -convex function restricted to a M -convex set is not M -convex, and (ii) show that this indeed means that Murota’s algorithm for M -convex function minimization is not provably optimal in our setting.

$$\beta_{X(t)}(d, b) := \max\{0, \min\{d + b, \beta_{X(t-1)} + X_t\}\}, \quad \beta_{X(0)}(d, b) = b$$

as a shorthand for the number of open docks and available bikes after the first t customers.

Our cost function is based on the number of out-of-stock events. In accordance with the above-described model, customer t experiences an out-of-stock event if and only if $\delta_{X(t)}(d, b) = \delta_{X(t-1)}(d, b)$. Since $d+b = \delta_{X(t)}(d, b) + \beta_{X(t)}(d, b)$ for every t , this happens if and only if $\beta_{X(t)}(d, b) = \beta_{X(t-1)}(d, b)$. As we are interested in the number of out-of-stock events as a function of the initial number of open docks and available bikes, we can write our cost-function $c^{X(t)}(d, b)$

$$= |\{\tau : \tau \leq t, X_\tau = 1, \delta_{X(\tau-1)}(d, b) = 0\}| + |\{\tau : \tau \leq t, X_\tau = -1, \beta_{X(\tau-1)}(d, b) = 0\}|.$$

It is then easy to see that with $c^{X(0)}(d, b) = 0$, $c^{X(t)}(d, b)$ fulfills the recursion

$$c^{X(t)}(d, b) = c^{X(t-1)}(d, b) + \mathbf{1}_{\{\beta_{X(t)}(d, b) = \beta_{X(t-1)}(d, b)\}}.$$

Given for each station $i \in [n]$ a distribution, which we call *demand-profile*, p_i over $\{(\pm 1)^s, s \in \mathbb{N}\}$, we can then write $c_i(d, b) = \mathbb{E}_{X \sim p_i}[c^X(d, b)]$ for the expected number of out-of-stock events at station i and $c(\mathbf{d}, \mathbf{b}) = \sum_i c_i(d_i, b_i)$. We then want to solve, for parameters $D, B, (\bar{\mathbf{d}}, \bar{\mathbf{b}})$, and z , as well as l_i, u_i for each $i \in [n]$, the following minimization problem

$$\begin{aligned} \text{minimize}_{(\mathbf{d}, \mathbf{b})} \quad & \sum_i c_i(d_i, b_i) \\ \text{s.t.} \quad & \sum_i d_i + b_i \leq D + B, \\ & \sum_i b_i \leq B, \\ & \sum_i |(\bar{d}_i + \bar{b}_i) - (d_i^* + b_i^*)| \leq z, \\ \forall i \in [n] : \quad & l_i \leq d_i + b_i \leq u_i. \end{aligned}$$

Here, the first constraint corresponds to a budget on the number of docks, the second to a budget on the number of bikes, the third to the operational constraints and the fourth to the lower and upper bound on the number of docks at each station. We assume without loss of generality that there exists an optimal solution in which the second constraint holds with equality; to ensure that, we may add a dummy ("depot") station \mathcal{D} that has $c_{\mathcal{D}}(\cdot, \cdot) = 0$, $l_{\mathcal{D}} = u_{\mathcal{D}} = B$, and run the algorithm with the budget on docks $(D + B)$ increased by B .

In Section 3 we prove that $c^X(\cdot, \cdot)$ fulfills a particular set of inequalities making it a so-called multimodular function.

Definition 1. [13],[1] A function $f : \mathbb{N}_0^2 \rightarrow \mathbb{R}$ is called multimodular if

$$f(d+1, b+1) - f(d+1, b) \geq f(d, b+1) - f(d, b); \quad (1)$$

$$f(d-1, b+1) - f(d-1, b) \geq f(d, b) - f(d, b-1); \quad (2)$$

$$f(d+1, b-1) - f(d, b-1) \geq f(d, b) - f(d-1, b); \quad (3)$$

for all d, b such that all terms are well-defined.

For future reference, we define the following additional inequalities, which are implied² by the above:

$$f(d+2, b) - f(d+1, b) \geq f(d+1, b) - f(d, b); \quad (4)$$

$$f(d, b+2) - f(d, b+1) \geq f(d, b+1) - f(d, b); \quad (5)$$

$$f(d+1, b+1) - f(d, b+1) \geq f(d+1, b) - f(d, b). \quad (6)$$

Even though we are motivated by the cost-functions defined in this section, our main results hold for arbitrary sums of such two-dimensional functions.

3 Multimodularity & An Allocation Algorithm

We first prove that the cost-functions defined in Section 2 are multimodular.

Lemma 2. $c^X(\cdot, \cdot)$ is multimodular for all X .

Proof. We prove the lemma by induction, showing that $X(t)$ is multimodular for all t . With $t = 0$, by definition, $c^{X(t)}(\cdot, \cdot) = 0$ and thus there is nothing to show. Suppose that $c^{X(0)}(\cdot, \cdot)$ through $c^{X(t-1)}(\cdot, \cdot)$ are all multimodular. We prove that $c^{X(t)}(\cdot, \cdot)$ is then multimodular as well.

We begin by proving inequality (1). Notice first that if

$$\max\{c^{X(1)}(d+1, b+1), c^{X(1)}(d+1, b), c^{X(1)}(d, b+1), c^{X(1)}(d, b)\} = 0,$$

we can use that inequality (1), by inductive assumption, holds after $t-1$ customers. Else, we use the inductive assumption on inequality (4) and (5) to prove inequality (1). If $X_1 = 1$ (and $d = 0$), then both sides of the inequality are 0 and $\delta_{X(1)}(d+1, b+1) = 0$, $\delta_{X(1)}(d+1, b) = 0$, $\delta_{X(1)}(d, b+1) = 0$, and $\delta_{X(1)}(d, b) = 0$. In that case, we may use inequality (5) (by induction) applied to the remaining $t-1$ customers. If instead $X_1 = -1$ (and $b = 0$), then both sides of the inequality are -1 and we have $\delta_{X(1)}(d+1, b+1) = d+b+2$, $\delta_{X(1)}(d+1, b) = d+b+1$, $\delta_{X(1)}(d, b+1) = d+b+1$, and $\delta_{X(1)}(d, b) = d+b$, so we may apply inequality (4) inductively to the remaining $t-1$.

It remains to prove inequalities (2) and (3). We restrict ourselves to inequality (2) as the proof for inequality (3) is symmetric with each X_i replaced by $-X_i$ and the coordinates of each term exchanged. As before, if

$$\max\{c^{X(1)}(d-1, b+1), c^{X(1)}(d-1, b), c^{X(1)}(d, b), c^{X(1)}(d, b-1)\} = 0,$$

the inductive assumption applies. If instead $X_1 = 1$ and the maximum is positive, then the LHS and the RHS are both 0 and we have $\delta_{X(1)}(d-1, b+1) = 0$, $\delta_{X(1)}(d-1, b) = 0$, $\delta_{X(1)}(d, b) = 0$, $\delta_{X(1)}(d, b-1) = 0$. In that case, both sides of the inequality are subsequently coupled and the inequality holds with equality.

In contrast, if $X_1 = -1$ and the maximum is positive, then $b = 1$, the RHS is -1, and the LHS is 0. In this case we have $\delta_{X(1)}(d-1, b+1) = d$, $\delta_{X(1)}(d-1, b) = d$, $\delta_{X(1)}(d, b) = d+1$, $\delta_{X(1)}(d, b-1) = d$. Let \hat{t} denote the next customer such that one of the four terms changes.

² (6) and (1) are equivalent, (1) and (2) imply (5), and (3) and (6) imply (4).

If $X_{\hat{t}} = 1$, then both terms on the LHS increase by 1, so it remains 0, whereas only the negative term on the RHS increases, so the inequality holds with $0 \geq -2$. Moreover, since $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = 0$, and $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = 0$; subsequently both sides of the inequality are again coupled.

Finally, if $X_{\hat{t}} = -1$, then both terms on the RHS increase by 1 with customer \hat{t} , but only the negative term on the LHS. Thus, thereafter both sides are again equal. In this case as well, both sides remain coupled thereafter since we have $\delta_{X(\hat{t})}(d-1, b+1) = \delta_{X(\hat{t})}(d, b) = d+b$, and $\delta_{X(\hat{t})}(d-1, b) = \delta_{X(\hat{t})}(d, b-1) = d+b-1$.

Recall that inequalities (4), (5), and (6) are implied by the first three, so this completes the proof. \square

Corollary 3. $c_i(\cdot, \cdot)$ is multimodular for any demand-profile p_i .

Proof. The proof is immediate from Lemma 2 and linearity of expectation. \square

3.1 An Allocation Algorithm

In this section, we present our algorithm. Intuitively, in each iteration the algorithm picks one dock and at most one bike within the system and moves them from one station to another. It chooses the dock, and the bike, so as to maximize the reduction in objective value. To formalize this notion, we define the *movement of a dock* via the following transformations.

Definition 4. We shall use the notation $(\mathbf{v}_{-i}, \hat{v}_i) := (v_1 \dots v_{i-1}, \hat{v}_i, v_{i+1} \dots v_n)$. Similarly, $(\mathbf{v}_{-i,-j}, \hat{v}_i, \hat{v}_j) := (v_1 \dots \hat{v}_i \dots \hat{v}_j \dots v_n)$. Then a dock-move from i to j corresponds to one of the following transformations of feasible solutions:

1. $o_{ij}(\mathbf{d}, \mathbf{b}) = ((\mathbf{d}_{-i,-j}, d_i - 1, d_j + 1), \mathbf{b})$ – Moving one open dock from i to j ;
2. $e_{ij}(\mathbf{d}, \mathbf{b}) = (\mathbf{d}, (\mathbf{b}_{-i,-j}, b_i - 1, b_j + 1))$ – Moving a dock & a bike from i to j ;
3. $E_{ijh}(\mathbf{d}, \mathbf{b}) = (\mathbf{d}_{-i,-h}, d_i - 1, d_h + 1), (\mathbf{b}_{-j,-h}, b_j + 1, b_h - 1))$ – Moving one open dock from i to j and one bike from h to j ;
4. $O_{ijh}(\mathbf{d}, \mathbf{b}) = (\mathbf{d}_{-j,-h}, d_j + 1, d_h - 1), (\mathbf{b}_{-i,-h}, b_i - 1, b_h + 1))$ – Moving one bike from i to h and one open dock from i to j .

Further, we define the neighborhood $N(\mathbf{d}, \mathbf{b})$ of (\mathbf{d}, \mathbf{b}) as the set of allocations that are one dock-move away from (\mathbf{d}, \mathbf{b}) . Formally,

$$N(\mathbf{d}, \mathbf{b}) := \{o_{ij}(\mathbf{d}, \mathbf{b}), e_{ij}(\mathbf{d}, \mathbf{b}), E_{ijh}(\mathbf{d}, \mathbf{b}), O_{ijh}(\mathbf{d}, \mathbf{b}) : i, j, h \in [n]\}.$$

Finally, define the dock-move distance between (\mathbf{d}, \mathbf{b}) and $(\mathbf{d}', \mathbf{b}')$ as

$$\sum_i |(d_i + b_i) - (d'_i + b'_i)|.$$

This gives rise to a very simple algorithm: we first find the optimal allocation of bikes for the current allocation of docks; the convexity of each c_i in the number of bikes, with fixed number of docks, implies that this can be done greedily by taking out all the bikes and then adding them one by one. Then, while there

exists a dock-move that improves the objective, we find the best possible such dock-move and update the allocation accordingly (cf. Algorithm 1).

Remark: each iteration of the algorithm can be implemented in (amortized) $O(T + \log(n))$ time by maintaining six binary heaps that contain for each change a dock-move could have at each station (i.e., add/take an open dock, add/take a bike, and add/take a bike and a dock) the change in objective this would yield. Instead of comparing all $O(n^2)$ possible moves, one can then find the argument of the minimum and update (\mathbf{d}, \mathbf{b}) in constant time, and then update the lists (for the stations involved in the dock-move) in $O(T + \log(n))$.

Algorithm 1 Greedy

```

1: Find optimal allocation of bikes for current dock allocation
2: while  $c(\mathbf{d}, \mathbf{b}) > \min_{(\mathbf{d}', \mathbf{b}') \in N(\mathbf{d}, \mathbf{b})} c(\mathbf{d}', \mathbf{b}')$  do
3:    $(\mathbf{d}, \mathbf{b}) \leftarrow \arg \min_{(\mathbf{d}', \mathbf{b}') \in N(\mathbf{d}, \mathbf{b})} c(\mathbf{d}', \mathbf{b}')$ 
4: end while

```

3.2 Proof of Optimality

We prove that the algorithm returns an optimal solution by showing that the condition in the while-loop is false only if (\mathbf{d}, \mathbf{b}) minimizes the objective; else, the algorithm moves a dock to find a better solution. Thus, if the algorithm terminates, then the solution is optimal. Before we prove Lemma 7 to establish this, we first define an allocation of bikes and docks as bike-optimal if it minimizes the objective among allocations with the same number of docks at each station and prove that bike-optimality is an invariant of the while-loop.

Definition 5. We call an allocation (\mathbf{d}, \mathbf{b}) bike-optimal if

$$(\mathbf{d}, \mathbf{b}) \in \arg \min_{(\hat{\mathbf{d}}, \hat{\mathbf{b}}): \forall i, d_i + b_i = \hat{d}_i + \hat{b}_i, \sum_i \hat{b}_i = B} \{c(\hat{\mathbf{d}}, \hat{\mathbf{b}})\}.$$

Lemma 6. Suppose (\mathbf{d}, \mathbf{b}) is bike-optimal. Given i and j , one of the possible dock-moves from i to j , i.e., $e_{ij}(\mathbf{d}, \mathbf{b})$, $o_{ij}(\mathbf{d}, \mathbf{b})$, $E_{ijh}(\mathbf{d}, \mathbf{b})$, or $O_{ijh}(\mathbf{d}, \mathbf{b})$, is bike-optimal, i.e., when moving a dock from i to j , one has to move at most one bike within the system to maintain bike-optimality.

Proof. It is known that multimodular functions fulfill certain convexity properties (see e.g., [22], [29]); in particular, for fixed d and b it is known that $c_i(k, d + b - k)$ is a convex function of $k \in \{0, \dots, d + b\}$. Thus, if $T_{ijh}(\mathbf{d}, \mathbf{b})$ was not bike-optimal, there would have to be two stations such that moving a bike from one to the other improves the objective. By the bike-optimality of (\mathbf{d}, \mathbf{b}) , at least one of these two stations must have been involved in T_{ijh} . We prove that the result holds if e_{ij} was the best of the set of possible moves $\{e_{ij}, o_{ij}, E_{ijh}, O_{ijh}\}_{i,j,h \in [n]}$ – the other three cases are almost symmetric. Let ℓ denote a generic third station. Then a bike improving the objective could correspond to one being moved from ℓ to j , from i to j , from i to ℓ , from ℓ to i , from j to ℓ or from j to i . In this case, a move from ℓ to j , i to j and i to ℓ yield the

allocations $E_{ij\ell}(\mathbf{d}, \mathbf{b})$, $o_{ij}(\mathbf{d}, \mathbf{b})$ and $O_{ij\ell}(\mathbf{d}, \mathbf{b})$, respectively. Since e_{ij} is assumed to be the minimizer among the possible dock-moves, none of these have objective smaller than that of $e_{ij}(\mathbf{d}, \mathbf{b})$. It remains to show that moving a bike from ℓ to i , j to ℓ or j to i yields no improvement. These all follow from bike-optimality of (\mathbf{d}, \mathbf{b}) and the multimodular inequalities. Specifically, an additional bike at i yields less improvement and a bike fewer at j has greater cost in $e_{ij}(\mathbf{d}, \mathbf{b})$ than in (\mathbf{d}, \mathbf{b}) , since

$$\begin{aligned} c_i(d_i - 1, b_i) - c_i(d_i - 2, b_i + 1) &\leq c_i(d_i, b_i) - c_i(d_i - 1, b_i + 1) \\ c_j(d_j + 2, b_j - 1) - c_j(d_j + 1, b_j) &\geq c_j(d_j + 1, b_j - 1) - c_j(d_j, b_j). \end{aligned}$$

Both of the above inequalities follow from inequality (3). \square

We are now ready to prove that when the algorithm terminates it must have found an optimal solution.

Lemma 7 (Neighborhood). *Suppose (\mathbf{d}, \mathbf{b}) is bike-optimal, but does not minimize $c(\cdot, \cdot)$ subject to budget constraints. Let $(\mathbf{d}^*, \mathbf{b}^*)$ denote a feasible solution with better objective at minimal dock-distance from (\mathbf{d}, \mathbf{b}) . As (\mathbf{d}, \mathbf{b}) is bike-optimal, there exist j and k such that $b_j + d_j < b_j^* + d_j^*$ and $b_k + d_k > b_k^* + d_k^*$. Pick any such j and k ; then there exists a dock-move to j or a dock-move from k that improves the objective of (\mathbf{d}, \mathbf{b}) .*

Proof. The proof of the lemma follows a a case-by-case analysis, each of which resembles the same idea: $(\mathbf{d}^*, \mathbf{b}^*)$ minimizes the dock-move distance to (\mathbf{d}, \mathbf{b}) among solutions with lower function value than (\mathbf{d}, \mathbf{b}) , i.e., among all $(\mathbf{d}^*, \mathbf{b}^*)$ such that $\sum_i d_i + b_i = \sum_i d_i^* + b_i^*$, $\sum_i b_i = \sum_i b_i^*$, and $c(\mathbf{d}^*, \mathbf{b}^*) < c(\mathbf{d}, \mathbf{b})$, $(\mathbf{d}^*, \mathbf{b}^*)$ has minimum dock-move distance to (\mathbf{d}, \mathbf{b}) . We show that with j and k as in the statement of the lemma, either there exists a dock-move to j /from k that improves the objective or there exists a solution $(\mathbf{d}^{**}, \mathbf{b}^{**})$ with objective value lower than (\mathbf{d}, \mathbf{b}) , $\sum_i d_i + b_i = \sum_i d_i^{**} + b_i^{**}$, $\sum_i b_i = \sum_i b_i^{**}$ and smaller dock-move distance to (\mathbf{d}, \mathbf{b}) . Since the latter contradicts our choice of $(\mathbf{d}^*, \mathbf{b}^*)$, this proves, that in (\mathbf{d}, \mathbf{b}) there must be a dock-move to j /from k that yields a lower objective. We distinguish among the following cases:

1. $d_j < d_j^*$ and $d_k > d_k^*$;
2. $b_j < b_j^*$ and $b_k > b_k^*$;
3. $d_j < d_j^*$, $b_k > b_k^*$, and $b_j \geq b_j^*$
 - (a) and there exists ℓ with $d_\ell + b_\ell \geq d_\ell^* + b_\ell^*$, $b_\ell < b_\ell^*$;
 - (b) and there exists ℓ with $d_\ell + b_\ell < d_\ell^* + b_\ell^*$, $b_\ell < b_\ell^*$;
 - (c) for all $\ell \notin \{j, k\}$, we have $b_\ell \geq b_\ell^*$, so $\sum_i b_i > \sum_i b_i^*$;
4. $b_j < b_j^*$, $d_j \geq d_j^*$, $b_k \leq b_k^*$ and $d_k > d_k^*$,
 - (a) and there exists ℓ with $d_\ell + b_\ell > d_\ell^* + b_\ell^*$ and $b_\ell > b_\ell^*$;
 - (b) and there exists ℓ with $d_\ell + b_\ell \leq d_\ell^* + b_\ell^*$ and $b_\ell > b_\ell^*$;
 - (c) for all $\ell \notin \{j, k\}$, we have $b_\ell \leq b_\ell^*$, so $\sum_i b_i < \sum_i b_i^*$.

We show that in case (1) a move from k to j yields improvement. The proof for case (2) is symmetric. Thus, in cases (3a) and (4a) there exists a move from k to ℓ , respectively from ℓ to j , that yields improvement. Since the proofs for cases (3b) and (4b) are also symmetric, we only present the proofs for (3b). Cases (3c) and (4c) contradict our assumption that $\sum_i b_i = \sum_i b_i^*$ and can thus be excluded.

For case (1), we define $(\mathbf{d}^{**}, \mathbf{b}^{**}) = e_{jk}(\mathbf{d}^*, \mathbf{b}^*)$, so

$$\begin{aligned} c((\mathbf{d}^{**}, \mathbf{b}^{**})) - c((\mathbf{d}^*, \mathbf{b}^*)) &= (c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*) + c_k(d_k^* + 1, b_k^*) - c_k(d_k^*, b_k^*)). \\ \text{Given that } \sum_i |d_i - d_i^*| + |b_i - b_i^*| &> \sum_i |d_i - d_i^{**}| + |b_i - b_i^{**}|, \text{ the definition of } (\mathbf{d}^*, \mathbf{b}^*) \text{ implies that this difference must be positive. Setting } (\mathbf{d}', \mathbf{b}') = e_{kj}(\mathbf{d}, \mathbf{b}), \\ c((\mathbf{d}, \mathbf{b})) - c((\mathbf{d}', \mathbf{b}')) &= \left(c_j(d_j, b_j) - c_j(d_j + 1, b_j) \right) + \left(c_k(d_k, b_k) - c_k(d_k - 1, b_k) \right) \\ &\geq \left(c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*) \right) + \left(c_k(d_k^* + 1, b_k^*) - c_k(d_k^*, b_k^*) \right) \\ &= c(\mathbf{d}^{**}, \mathbf{b}^{**}) - c(\mathbf{d}^*, \mathbf{b}^*) > 0. \end{aligned}$$

We prove the inequality between the second and third expression by showing that it holds for each difference, starting with

$$c_j(d_j, b_j) - c_j(d_j + 1, b_j) \geq c_j(d_j^* - 1, b_j^*) - c_j(d_j^*, b_j^*).$$

Applying inequality (3) given in the definition of multimodularity, $t \geq 0$ times bounds the right-hand side (RHS) by $c_j(d_j^* - 1 - t, b_j^* + t) - c_j(d_j^* - t, b_j^* + t)$. Setting $t = d_j^* - d_j - 1 \geq 0$, we then find that the RHS is bounded above by

$$c_j(d_j, b_j^* + d_j^* - d_j - 1) - c_j(d_j + 1, b_j^* + d_j^* - d_j - 1).$$

On the other hand, applying inequality (6) repeatedly to the left-hand side (LHS) shows that $\forall s \geq 0$, the LHS is at least $c_j(d_j, b_j + s) - c_j(d_j + 1, b_j + s)$. Hence, by setting $s = b_j^* + d_j^* - d_j - b_j - 1$, which is non-negative since $b_j + d_j < b_j^* + d_j^*$, we bound the LHS from below by

$$c_j(d_j, b_j + b_j^* + d_j^* - d_j - b_j - 1) - c_j(d_j + 1, b_j + b_j^* + d_j^* - d_j - b_j - 1).$$

This equals the upper bound on the RHS and thus proves the desired inequality. Similarly, to show that

$$c_k(d_k - 1, b_k) - c_k(d_k, b_k) \leq c_k(d_k^*, b_k^*) - c_k(d_k^* + 1, b_k^*), \quad (7)$$

we apply inequality (3) $d_k - d_k^* - 1$ times to bound the LHS in (7) by $c_k(d_k^*, b_k + d_k - d_k^* + 1) - c_k(d_k^* + 1, b_k + d_k - d_k^* + 1)$. Thereafter, we apply inequality (5) $b_k + d_k - d_k^* + 1 - b_k' \geq 0$ times to obtain the desired bound.

In case (3b), we define $(\mathbf{d}^{**}, \mathbf{b}^{**}) = E_{j\ell}(\mathbf{d}^*, \mathbf{b}^*)$ and $(\mathbf{d}', \mathbf{b}') = O_{kj\ell}(\mathbf{d}, \mathbf{b})$. Similarly to the first case, we need to show that $c((\mathbf{d}, \mathbf{b})) - c((\mathbf{d}', \mathbf{b}')) \geq c(\mathbf{d}^{**}, \mathbf{b}^{**}) - c(\mathbf{d}^*, \mathbf{b}^*)$. Since all terms not involving j, k , and ℓ cancel out and the terms involving j and k can be bounded the same way as before, deriving

$$c_\ell(d_\ell, b_\ell) - c_\ell(d_\ell - 1, b_\ell + 1) \geq c_\ell(d_\ell^* + 1, b_\ell^* - 1) - c_\ell(d_\ell^*, b_\ell^*)$$

suffices. We obtain this inequality by repeatedly applying inequalities (3) and (4) to the LHS. \square

4 Operational Constraints & Running Time

In this section, we show that the allocation algorithm is optimal for the operational constraints introduced in Section 2 and thereby also provide an analysis of the running-time of the algorithm. To do so, we first define the set of feasible solutions with respect to those constraints.

Definition 8. Define the z -dock ball $S_z(\mathbf{d}, \mathbf{b})$ around (\mathbf{d}, \mathbf{b}) as the set of allocations with dock-move distance at most $2z$, i.e., $S_0(\mathbf{d}, \mathbf{b}) = \{(\mathbf{d}, \mathbf{b})\}$ and

$$S_z(\mathbf{d}, \mathbf{b}) = S_{z-1}(\mathbf{d}, \mathbf{b}) \cup \left(\bigcup_{(\mathbf{d}', \mathbf{b}') \in S_{z-1}(\mathbf{d}, \mathbf{b})} N(\mathbf{d}', \mathbf{b}') \right).$$

We now want to prove that Lemma 7 continues to hold in the constrained setting; in particular, we show that even with the operational constraints, local optima are global optima.

Lemma 9 (z -step neighborhood). If $(\hat{\mathbf{d}}, \hat{\mathbf{b}}) \in S_z(\mathbf{d}, \mathbf{b}) \setminus S_{z-1}(\mathbf{d}, \mathbf{b})$ is bike-optimal and $c(\mathbf{d}^*, \mathbf{b}^*) < c(\hat{\mathbf{d}}, \hat{\mathbf{b}})$ for some $(\mathbf{d}^*, \mathbf{b}^*) \in S_z(\mathbf{d}, \mathbf{b}) \setminus S_{z-1}(\mathbf{d}, \mathbf{b})$, then there exists $(\mathbf{d}', \mathbf{b}') \in S_z(\mathbf{d}, \mathbf{b}) \cap N(\hat{\mathbf{d}}, \hat{\mathbf{b}})$ such that $c(\mathbf{d}', \mathbf{b}') < c(\hat{\mathbf{d}}, \hat{\mathbf{b}})$.

Proof. Notice that this lemma closely resembles Lemma 7: the sole difference lies in Lemma 7 not enforcing the dock-move to maintain a bound on the distance to some allocation (\mathbf{d}, \mathbf{b}) .

Define $(\mathbf{d}^*, \mathbf{b}^*)$ as in Lemma 7 with the additional restriction that $(\mathbf{d}^*, \mathbf{b}^*)$ be in $S_z(\mathbf{d}, \mathbf{b})$, i.e., pick a solution in $S_z(\mathbf{d}, \mathbf{b})$ that minimizes the dock-move distance to $(\hat{\mathbf{d}}, \hat{\mathbf{b}})$ among solutions with strictly smaller objective value. We argue again that bike-optimality of $(\hat{\mathbf{d}}, \hat{\mathbf{b}})$ implies that there exist j and k , such that $\hat{d}_j + \hat{b}_j < d_j^* + b_j^*$, and $\hat{d}_k + \hat{b}_k > d_k^* + b_k^*$. Further, for any such j and k , we can apply the proof of Lemma 7 to find a move involving at least one of the two that decreases both the objective value and the dock-move distance to $(\mathbf{d}^*, \mathbf{b}^*)$.

We aim to find j and k such that the move identified, say from ℓ to m , is guaranteed to remain within $S_z(\mathbf{d}, \mathbf{b})$. Notice that $|\{j\} \cap \{m\}| + |\{k\} \cap \{\ell\}| \geq 1$. We know that $d_m^* + b_m^* > \hat{d}_m + \hat{b}_m$ and $d_\ell^* + b_\ell^* < \hat{d}_\ell + \hat{b}_\ell$. Suppose the move from ℓ to m yields a solution outside of $S_z(\mathbf{d}, \mathbf{b})$. It follows that $\hat{d}_m + \hat{b}_m \geq d_m + b_m$ and $\hat{d}_\ell + \hat{b}_\ell \leq d_\ell + b_\ell$, so in particular either $\hat{d}_j + \hat{b}_j \geq d_j + b_j$ or $\hat{d}_k + \hat{b}_k \leq d_k + b_k$. Thus, if we can identify j and k such that those two inequalities do not hold, we are guaranteed that the identified move remains within $S_z(\mathbf{d}, \mathbf{b})$.

Define

$$k := \arg \max_i \{ \hat{d}_i + \hat{b}_i - \max\{d_i + b_i, d_i^* + b_i^*\} \}$$

We can then write

$$\begin{aligned} & \max_i \{ \hat{d}_i + \hat{b}_i - \max\{d_i + b_i, d_i^* + b_i^*\} \} \geq \\ & \min_i \left\{ 1, \max_{i: \hat{d}_i + \hat{b}_i > d_i + b_i} \{ (\hat{d}_i + \hat{b}_i) - (d_i^* + b_i^*) \} \right\} \end{aligned}$$

The latter is at least 1 unless it is the case for all i that if $\hat{d}_i + \hat{b}_i > d_i + b_i$ then $\hat{d}_i + \hat{b}_i \leq d_i^* + b_i^*$. Thus, unless the above condition fails, we have identified a k with the required properties. Suppose the condition does fail. Then

$$2z = \sum_i |(d_i + b_i) - (d_i^* + b_i^*)| = \sum_i |(d_i + b_i) - (\hat{d}_i + \hat{b}_i)|$$

$$\text{and } \sum_i d_i + b_i = \sum_i d_i^* + b_i^* = \sum_i \hat{d}_i + \hat{b}_i$$

imply that for all i with $\max\{\hat{d}_i + \hat{b}_i, d_i^* + b_i^*\} > d_i + b_i$, we have $\hat{d}_i + \hat{b}_i = d_i^* + b_i^*$. Thus, it must be the case that m fulfills $\hat{d}_m + \hat{b}_m < d_m + b_m$.

The argument for j is symmetric. \square

Theorem 10. *Starting with a bike-optimal allocation (\mathbf{d}, \mathbf{b}) , in the z -th iteration, the greedy algorithm finds an optimal allocation among those in $S_z(\mathbf{d}, \mathbf{b})$.*

Proof. We prove the theorem by induction in z . The base-case $z = 0$ holds trivially. Suppose in the z th iteration, the greedy algorithm has found the allocation $(\mathbf{d}^z, \mathbf{b}^z) \in \arg \min_{(\mathbf{d}^*, \mathbf{b}^*) \in S_z(\mathbf{d}, \mathbf{b})} c(\mathbf{d}^*, \mathbf{b}^*)$. We need to show that

$$(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) := \arg \min_{(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \in N(\mathbf{d}^z, \mathbf{b}^z)} \{c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})\}$$

minimizes the cost function among solutions in $S_{z+1}(\mathbf{d}, \mathbf{b})$.

We first observe that by Lemma 9, it suffices to show that there is no better solution in $S_{z+1}(\mathbf{d}, \mathbf{b})$ that is just one dock-move away from $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})$. Further, by Lemma 6 and the choice of dock-moves in the greedy algorithm we know that $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})$ must be bike-optimal. Let i be the station from which a dock was moved and let j be the station to which it was moved in the $z + 1$ st iteration. We denote a third station by h if the $z + 1$ st move involved a third one (recall that a dock-move from i to j can take an additional bike from i to a third station h or take one from h to j). We can then immediately exclude the following cases:

1. Any dock-move in which i receives a dock from some station ℓ , including possibly $\ell = j$ or $\ell = h$, can be excluded since the greedy algorithm could have chosen to take a dock from ℓ instead of i and found a bike-optimal allocation (by Lemma 6).
2. The same holds for any dock-move in which a dock is taken from j .
3. A dock-move not involving either of i , j , and h yields the same improvement as it would have prior to the $z + 1$ st iteration. Furthermore, if such a dock-move yields a solution within $S_{z+1}(\mathbf{d}, \mathbf{b})$, then prior to the $z + 1$ st iteration it would have yielded a solution within $S_z(\mathbf{d}, \mathbf{b})$. Hence, by the induction assumption, it cannot yield any improvement.
4. A dock-move from station i (or to j), as is implied by the fourth, fifth, and sixth inequality in the definition of multimodularity increases the objective at i more (decreases the objective at j less) than it would have prior to the $z + 1$ st iteration.

We are left with a dock-move from or to h as well as dock-moves that involve one of the three stations only via a bike being moved. Suppose that the dock-move in iteration $z + 1$ was E_{ijh} ; the case of O_{ijh} is symmetric. In this case, a subsequent move of a dock and a bike from h , i.e., $o_{h\ell}$ or $O_{h\ell m}$ for some m , increases the objective at h by at least as much as it did before (by inequality (2)) and can thus be excluded. The same holds for the move of an empty dock to h (by inequality (3)).

However, subsequent moves of an empty dock from h (or a full dock to h) have a lower cost (greater improvement) and require a more careful argument. Suppose $e_{h\ell}$ yielded an improvement – the cases for $E_{h\ell m}$, $o_{\ell h}$, and $E_{\ell hm}$ are similar. Notice first that if it were the case that $d_h^z + b_h^z > d_h + b_h$ and $d_\ell^z + b_\ell^z < d_\ell + b_\ell$, then $e_{h\ell}(E_{ijh}(\mathbf{d}^z, \mathbf{b}^z)) \in S_z(\mathbf{d}, \mathbf{b})$ and has a lower objective than $(\mathbf{d}^z, \mathbf{b}^z)$ which contradicts the inductive assumption. Furthermore, since it must be the case that $e_{h\ell}(E_{ijh}(\mathbf{d}^z, \mathbf{b}^z)) \in S_{z+1}(\mathbf{d}, \mathbf{b}) \setminus S_z(\mathbf{d}, \mathbf{b})$, it must also follow that either

1. $d_h^z + b_h^z > d_h + b_h$ and $d_\ell^z + b_\ell^z \geq d_\ell + b_\ell$ or
2. $d_h^z + b_h^z \leq d_h + b_h$ and $d_\ell^z + b_\ell^z < d_\ell + b_\ell$,

since otherwise a dock-move from h to ℓ would either yield a solution in S_z or one not in S_{z+1} . Notice further that the inductive assumption implies that $(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \notin S_z(\mathbf{d}, \mathbf{b})$. Thus, it must be the case that $d_i^{z+1} + b_i^{z+1} < d_i + b_i$ and $d_j^{z+1} + b_j^{z+1} < d_j + b_j$. We can thus argue in the following way about

$$\begin{aligned} & c(e_{h\ell}(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})) - c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) = \\ & c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1) + c_\ell(d_\ell^z + 1, b_\ell^z) - c_\ell(d_\ell^z, b_\ell^z). \end{aligned}$$

In the first case, since $o_{h\ell}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$, the inductive assumption implies that $c_h(d_h^z, b_h^z - 1) + c_j(d_j^z, b_j^z + 1) \geq c_h(d_h^z, b_h^z) + c_j(d_j^z, b_j^z)$. Further, by the choice of the greedy algorithm, an additional empty dock at ℓ has no more improvement than an additional dock and an additional bike at j minus the cost of taking the bike from h ; otherwise, the greedy algorithm would have moved an empty dock from h to ℓ in the $z + 1$ st iteration. Thus,

$$\begin{aligned} & c_\ell(d_\ell^z + 1, b_\ell^z) - c_\ell(d_\ell^z, b_\ell^z) \\ & \leq c_j(d_j^z, b_j^z) - c_j(d_j^z, b_j^z + 1) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z) \\ & \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z, b_h^z) - c_h(d_h^z + 1, b_h^z - 1) + c_h(d_h^z, b_h^z) \\ & \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1), \end{aligned}$$

implying that $c(e_{h\ell}(\mathbf{d}^{z+1}, \mathbf{b}^{z+1})) - c(\mathbf{d}^{z+1}, \mathbf{b}^{z+1}) \geq 0$.

In the second case, since we know that $e_{i\ell}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$, the inductive assumption implies $c_\ell(d_\ell^z + 1, b_\ell^z) + c_i(d_i^z - 1, b_i^z) \geq c_\ell(d_\ell^z, b_\ell^z) + c_i(d_i^z, b_i^z)$. Further, the choice of the greedy algorithm to take the dock from i , not h , implies that $c_i(d_i^z, b_i^z) - c_i(d_i^z - 1, b_i^z) \leq c_h(d_h^z, b_h^z - 1) - c_h(d_h^z + 1, b_h^z - 1)$. Combining these two inequalities again implies that $e_{h\ell}$ does not yield an improvement.

The remaining cases are ones in which a move only involves i, j , or h as the third station that a bike is taken from/added to. Suppose that the transformation

in iteration $z + 1$ was E_{ijh} – the other cases are similar. A subsequent move of a bike to i (by inequality (3)) or j (by inequality (2)) yields at most the improvement that it would have prior to iteration $z + 1$. Same holds for taking a bike from h (by combining inequalities (2) and (3)). Thus, the remaining cases are those in which a bike is taken from i or j as well as the ones in which a bike is added to h .

For a bike taken from i , notice that the greedy choice was to take a bike from h rather than from i , so the increase in objective in taking it from i now is at least what it was at h in the $z + 1$ st iteration. Similarly, since the greedy algorithm chose E_{ijh} over e_{ij} , taking the bike from j has cost at least the cost it had prior to the $z + 1$ st iteration at h . But since $E_{\ell mh}$, for some ℓ and m for which it was feasible before the $z + 1$ st iteration, did not yield an improvement then, it follows that $E_{\ell mi}$ and $E_{\ell mj}$ do not yield an improvement after the $z + 1$ st iteration.

For a bike added to h , the argument is similar to the one about a dock taken from h after a bike was taken from h . For $O_{\ell mh}$ to be feasible, for some ℓ, m within S_{z+1} , it must be the case that either $d_m^z + b_m^z < d_m + b_m$ or $d_\ell^z + b_\ell^z > d_\ell + b_\ell$.

In the former case, $e_{im}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$, so the inductive assumption implies that the increase in cost of taking a dock from i in the $z + 1$ st iteration is at least the decrease realized by moving a dock to m . But the increase in objective in taking a dock and a bike from ℓ is at least the increase at i and h in the $z + 1$ st iteration, since otherwise the greedy algorithm would have taken the bike and dock from ℓ . Hence, the decrease in objective at h and at m is bounded above by the increase in objective at ℓ .

In the latter case, the inductive assumption implies that an increase in objective at ℓ due to $O_{\ell mh}$ is bounded below by the increase in objective prior to the $z + 1$ st iteration due to $o_{\ell j}$ (since $o_{\ell j}(\mathbf{d}^z, \mathbf{b}^z) \in S_z(\mathbf{d}, \mathbf{b})$). That improvement however is greater-equal to the decrease $O_{\ell mh}$ yields at m and at h combined by the choice of the greedy algorithm in iteration $z + 1$. Thus, $O_{\ell mh}$ cannot yield an improvement. \square

An immediate corollary of the above result yields a bound on the number of iterations the greedy algorithm may run for.

Corollary 11. *The greedy algorithm terminates in at most $\sum_i d_i + b_i$ iterations.*

Technically, we might view the size of the input as $\log(B + D)$; however, in our application the physical entities of docks and bikes are truly given in an unary encoding.

Conclusion Our work provides a fast and provably efficient algorithm for the problem of minimizing the sum of two-dimensional multimodular functions under constraints. It has strong connections to the literature on discrete convex analysis as well as more novel work on the optimization of bike-sharing systems.

References

1. Altman, E., Gaujal, B., Hordijk, A.: Multimodularity, convexity, and optimization properties. *Mathematics of Operations Research* 25(2), 324–347 (2000)
2. Capital Bikeshare: Capital Bikeshare member survey report (2014)
3. de Chardon, C.M., Caruso, G., Thomas, I.: Bike-share rebalancing strategies, patterns, and purpose. *Journal of Transport Geography* 55, 22–39 (2016)
4. Chemla, D., Meunier, F., Calvo, R.W.: Bike sharing systems: Solving the static rebalancing problem. *Discrete Optimization* 10(2), 120–146 (2013)
5. Chen, L., Zhang, D., Wang, L., Yang, D., Ma, X., Li, S., Wu, Z., Pan, G., Thi-Mai-Trang Nguyen, J.J.: Dynamic cluster-based over-demand prediction in bike sharing systems. In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. pp. 841–852. ACM (2016)
6. Datner, S., Raviv, T., Tzur, M., Chemla, D.: Setting inventory levels in a bike sharing network (2015)
7. Dell’Amico, M., Hadjicostantinou, E., Iori, M., Novellani, S.: The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. *Omega* 45, 7–19 (2014)
8. Erdoğan, G., Battarra, M., Calvo, R.W.: An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. *European Journal of Operational Research* 245(3), 667–679 (2015)
9. Erdoğan, G., Laporte, G., Calvo, R.W.: The static bicycle relocation problem with demand intervals. *European Journal of Operational Research* 238(2), 451–457 (2014)
10. Forma, I.A., Raviv, T., Tzur, M.: A 3-step math heuristic for the static repositioning problem in bike-sharing systems. *Transportation Research Part B: Methodological* 71, 230–247 (2015)
11. Freund, D., Norouzi-Fard, A., Paul, A., Henderson, S.G., Shmoys, D.B.: (even) smarter tools for (Citi)Bike sharing (2016), working paper
12. Ghosh, S., Trick, M., Varakantham, P.: Robust repositioning to counter unpredictable demand in bike sharing systems (2016)
13. Hajek, B.: Extremal splittings of point processes. *Mathematics of operations research* 10(4), 543–556 (1985)
14. Ho, S.C., Szeto, W.: Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review* 69, 180–198 (2014)
15. Jian, N., Freund, D., Wiberg, H., Henderson, S.G.: Simulation optimization for a large-scale bike-sharing system. *Winter Simulation Conference* (to appear) (2016)
16. Jian, N., Henderson, S.G.: An introduction to simulation optimization. In: *Proceedings of the 2015 Winter Simulation Conference*. pp. 1780–1794. IEEE Press (2015)
17. Kabra, A., Belavina, E., Girotra, K.: Bike-share systems: Accessibility and availability. *Chicago Booth Research Paper* (15-04) (2015)
18. Kaspi, M., Raviv, T., Tzur, M.: Bike sharing systems: User dissatisfaction in the presence of unusable bicycles (2015)
19. Lee, Y.T., Sidford, A., Wong, S.C.w.: A faster cutting plane method and its implications for combinatorial and convex optimization. In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. pp. 1049–1065. IEEE (2015)

20. Li, Y., Zheng, Y., Zhang, H., Chen, L.: Traffic prediction in a bike-sharing system. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. p. 33. ACM (2015)
21. Liu, J., Sun, L., Chen, W., Xiong, H.: Rebalancing bike sharing systems: A multi-source data smart optimization (2016)
22. Murota, K.: Discrete convex analysis. SIAM (2003)
23. Murota, K.: On steepest descent algorithms for discrete convex functions. SIAM Journal on Optimization 14(3), 699–707 (2004)
24. Nair, R., Miller-Hooks, E., Hampshire, R.C., Bušić, A.: Large-scale vehicle sharing systems: analysis of vélib’. International Journal of Sustainable Transportation 7(1), 85–106 (2013)
25. O’Mahony, E.: Smarter Tools For (Citi) Bike Sharing. Ph.D. thesis, Cornell University (2015)
26. O’Mahony, E., Shmoys, D.B.: Data analysis and optimization for (citi) bike sharing. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 687–694 (2015)
27. Parikh, P., Ukkusuri, S.V.: Estimation of optimal inventory levels at stations of a bicycle sharing system (2014)
28. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: A variable neighborhood search approach. In: European Conference on Evolutionary Computation in Combinatorial Optimization. pp. 121–132. Springer (2013)
29. Raviv, T., Kolka, O.: Optimal inventory management of a bike-sharing station. IIE Transactions 45(10), 1077–1093 (2013)
30. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal on Transportation and Logistics 2(3), 187–229 (2013)
31. Schuijbroek, J., Hampshire, R., van Hoes, W.J.: Inventory rebalancing and vehicle routing in bike sharing systems. European Journal of Operations Research (to appear)
32. Singhvi, D., Singhvi, S., Frazier, P.I., Henderson, S.G., OMahony, E., Shmoys, D.B., Woodard, D.B.: Predicting bike usage for New York citys bike sharing system. Citeseer (2015)
33. Wang, J., Tsai, C.H., Lin, P.C.: Applying spatial-temporal analysis and retail location theory to public bikes site selection in Taipei. Transportation Research Part A: Policy and Practice 94, 45–61 (2016)
34. Zhang, J., Pan, X., Li, M., Yu, P.S.: Bicycle-sharing system analysis and trip prediction. arXiv preprint arXiv:1604.00664 (2016)

5 Appendix: Connections to M -Convex Functions

In this appendix we first provide the definitions of M -convex sets and functions, and then show that our cost-function with budget constraints is not M -convex. For the definitions, it is useful to denote $\text{supp}^+(\mathbf{x} - \mathbf{y}) = \{i : x_i > y_i\}$, $\text{supp}^-(\mathbf{x} - \mathbf{y}) = \{i : x_i < y_i\}$, and \mathbf{e}_i as the canonical unit vector.

Definition 12 (M -convex set). A nonempty set of integer points $B \subseteq \mathbb{Z}^{2n}$ is defined to be an M -convex set if it satisfies the following exchange axiom:

$$\forall \mathbf{x}, \mathbf{y} \in B, i \in \text{supp}^+(\mathbf{x} - \mathbf{y}), \exists j \in \text{supp}^-(\mathbf{x} - \mathbf{y}) : \mathbf{x} - \mathbf{e}_i + \mathbf{e}_j \in B.$$

Definition 13 (M -convex function). A function f is M -convex if for all $x, y \in \text{dom}(f)$, $i \in \text{supp}^+(x - y)$, $\exists j \in \text{supp}^-(x - y)$:

$$f(x) + f(y) \geq f(x - e_i + e_j) + f(y + e_i - e_j).$$

Kaspi et al. [18] prove a statement equivalent to $c(\cdot, \cdot)$ being M -convex. Murota [23] characterized the minimum of a M convex function as follows:

Lemma 14. [22] For an M -convex function f and $x \in \text{dom}(f)$ we have

$$f(x) \leq f(y) \forall y \text{ if and only if } f(x) \leq f(x - e_i + e_j) \forall i, j.$$

Thus, Algorithm 2 minimizes M -convex functions:

Algorithm 2 M -convex function minimization [23]

- 1: Find a vector $x \in \text{dom}(f)$
 - 2: Find i, j that minimize $f(x - e_i + e_j)$
 - 3: If $f(x) > f(x - e_i + e_j)$, set $x := x - e_i + e_j$ and go to 2
 - 4: Else, return x
-

We now show that even though our objective function $c(\cdot, \cdot)$ is M -convex and the underlying feasible set (with budget constraints) is M -convex, the restriction of c to the underlying feasible set may not be.

Example 15. Our example consists of three stations i, j , and k with demand-profiles:

$$\begin{aligned} p_i(-1) &= \frac{1}{2}, p_i(+1, -1) = \frac{1}{2}; \\ p_j(+1) &= \frac{1}{2}; \\ p_k(+1, -1, -1) &= 1. \end{aligned}$$

We consider two solutions. In the first, i, j , and k each have a dock allocated with i also having a bike allocated, i.e., $b'_i = d'_j = d'_k = 1$, whereas $d'_i = b'_j = b'_k = 0$ and our budget constraint is $D = 2, B = 1$. Then $c_i(d'_i, b'_i) = \frac{1}{2}$, $c_j(d'_j, b'_j) = 0$, and $c_k(d'_k, b'_k) = 1$. In the second solution, $d_i^* = b_k^* = d_k^* = 1$, whereas $b_i^* = d_j^* = b_j^* = 0$. Thus, we have $c_i(d_i^*, b_i^*) = \frac{1}{2}$, $c_j(d_j^*, b_j^*) = \frac{1}{2}$, and $c_k(d_k^*, b_k^*) = 0$, giving that $1 = c(\mathbf{d}^*, \mathbf{b}^*) < c(\mathbf{d}', \mathbf{b}') = \frac{3}{2}$. But then the statement of Lemma 14 with $y = (\mathbf{d}^*, \mathbf{b}^*)$ and $x = (\mathbf{d}', \mathbf{b}')$ implies that, if c is M -convex one of the following must be strictly smaller than $c(\mathbf{d}', \mathbf{b}')$:

$$\begin{aligned} c((\mathbf{d}'_{-i}, d'_i + 1), (\mathbf{b}'_{-i}, b'_i - 1)) &= \frac{3}{2}; \\ c(\mathbf{d}', (\mathbf{b}'_{-i, -k}, b'_i - 1, b'_k + 1)) &= 2; \\ c((\mathbf{d}'_{-i, -j}, d'_i + 1, d'_j - 1), \mathbf{b}') &= 2. \end{aligned}$$

Since this is not the case, we find that c restricted to the feasible set is not M -convex, even though the underlying feasible set is M -convex.